**TEMPLEOS**

**Credit:** https://templeos.org

# Giving praise to the TempleOS

**Michael Reed** delves into one of the most unusual operating systems he's ever seen and explores the story behind its creation.

**OUR EXPERT**

**Michael Reed** is a Linux veteran and was raised on lesser-known operating systems such as RISC OS, AmigaOS and OS/2.

**QUICK TIP**

**There is no simple way to change the resolution or font size in TempleOS. To keep things as readable as possible when running TempleOS on a virtualiser, consider switching on screen scaling and stretching the display window of the virtualiser, at the cost of some sharpness.**

TempleOS and its modernised derivative ZealOS are unusual operating systems that don't fit into the modern paradigm of how OSes are expected to look or work. Every aspect of the system, from the external user interface to the nitty gritty of how it works on the inside, defies convention.

The man who created TempleOS, Terry A Davis, was affected by serious mental illness problems, and many of his design decisions about the OS were informed by his eccentric views, including his unusual religious beliefs. Throughout his adult life, he had run-ins with the law and had difficulty supporting himself. He often got into spats on online forums and websites. However, he single-handedly created a complete, if unusual, OS worthy of exploration. Sadly, he died in 2018.

There have been attempts to continue the legacy of TempleOS. ZealOS is a modern implementation, and it's another system that we're going to look at.

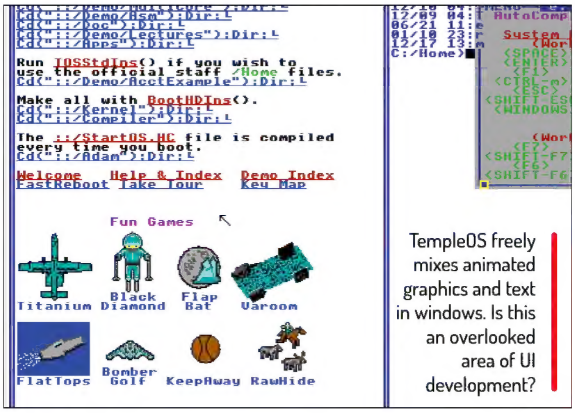
## Obtaining and installing

We'll start our investigation into TempleOS (**https://templeos.org**) in the usual way, by obtaining it and installing it. TempleOS is supplied as an ISO that boots to a live environment from which it is possible to install the operating system to a hard disk drive. On launching the ISO, you are given your first taste of the TempleOS aesthetic, which is loud, unconventional and largely text-based. Even at this early stage, things deviate from what you might be used to seeing in an installer, and it takes a minute to orientate yourself and start to navigate the interface.

When you've got over the sensory overload, it's time to choose the installation partition. If you answer in the affirmative when asked if you are installing in a VM, the installation wizard is slightly simplified, but we should warn you, this gives it something of a hair trigger when it comes to the destination partition, so be careful. Other than that, the installation is fairly quick and simple. At the end, you remove the installation media and accept the option to reboot, and the actual boot takes only a few seconds.

## First boot

The installation procedure of TempleOS should have given you a rough idea of what to expect from the user interface. We recommend saying yes to the offer of a tour. This takes the form of an interactive tutorial, and as TempleOS breaks with so many modern GUI conventions, you'd be unlikely to fully fathom the basics of the interface with experimentation alone. As an aside, interactive in-system tutorials have largely died out these days, having been replaced with video tutorials. You can revisit the help window that leads to these tutorials by pressing F1 at any time.



```
Cd("::/Demo/Asm");Dir;
Cd("::/Doc");Dir;
Cd("::/Demo/Lectures");Dir;
Cd("::/Apps");Dir;

Run IOSStdIns() if you wish to
use the official staff /Home files.
Cd("::/Demo/AcctExample");Dir;

Make all with BootHDIns().
Cd("::/Kernel");Dir;
Cd("::/Compiler");Dir;

The ::/StartOS.HC file is compiled
every time you boot.
Cd("::/Adam");Dir;
```

```
12/09 04:
06/21 11:
01/10 23:
12/17 13:
C:/Home>
```

TempleOS freely mixes animated graphics and text in windows. Is this an overlooked area of UI development?

In some respects, the TempleOS interface looks like a text mode display, but it supports simple 16-colour graphics as well as text. It's a multitasking system, with each task running inside its own window. As is the case with a typical desktop GUI operating system, these windows can be moved around with the mouse. The colour scheme is, frankly, a little garish. It reminds us of the text displays of the 8-bit era of home computers and also of some DOS-era PC software. In machines of that vintage, there sometimes wasn't much subtle variation in the colour palette, and it's obvious the creator of TempleOS was heavily influenced by that period of machines and software.

Much of the interface design approach unifies concepts that are normally kept separate in a GUI environment. For example, within a directory listing in a text window, the cursor can be moved around and files can be selected. It's also possible to click on files in such a listing with the mouse, meaning that a directory listing at the command line is also a file and program launcher. This seems unusual at first, but we're left wondering, would it be possible to add some extra interactivity with files on the Linux command line?

Taking things even further, graphics are sometimes thrown into the text environment. Some of these graphical elements can be interacted with, like icons in a GUI, and some are animated.

## Scripting and language

As is the case with most OSes, there is a connection between the scripting language and the command line. TempleOS takes this further and integrates the command line, scripting language and programming language. All of these use a C-like language that was created especially for TempleOS, called HolyC.

In C-style languages, you can call a function with a statement such as: `printf("Hello");`. For ease of use, the TempleOS command line allows slightly shortened commands, so `Dir;` lists the contents of the current directory. As it uses C-style syntax, you can combine commands on one line. For example, `Cd("/home");Dir;` changes to the **home** directory and lists its contents.

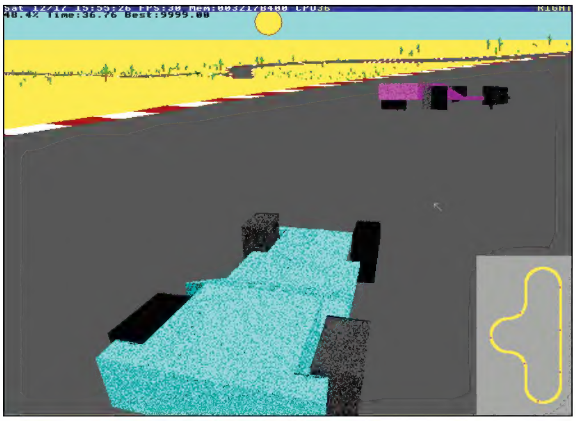
You might think this means that HolyC is a scripting language, but it's actually a compiled language like C. Behind the scenes, the TempleOS HolyC compiler interprets and compiles each line that is entered at the command line. One advantage of this approach is that it simplifies the process of learning the command line, scripting language and programming language.

Much of the system is implemented in this way. In most systems, utilities and programs are stored as object code (the result of compiling source code) or a scripting language, but TempleOS blurs the distinction as most programs and utilities are stored as source code and compiled and run as needed. One advantage is that the program source code can be edited at any time. It does mean program startup might be delayed, but it isn't noticeable for smaller programs. Unlike implementing programs using a scripting language, programs execute at full speed when running.

## Under the hood

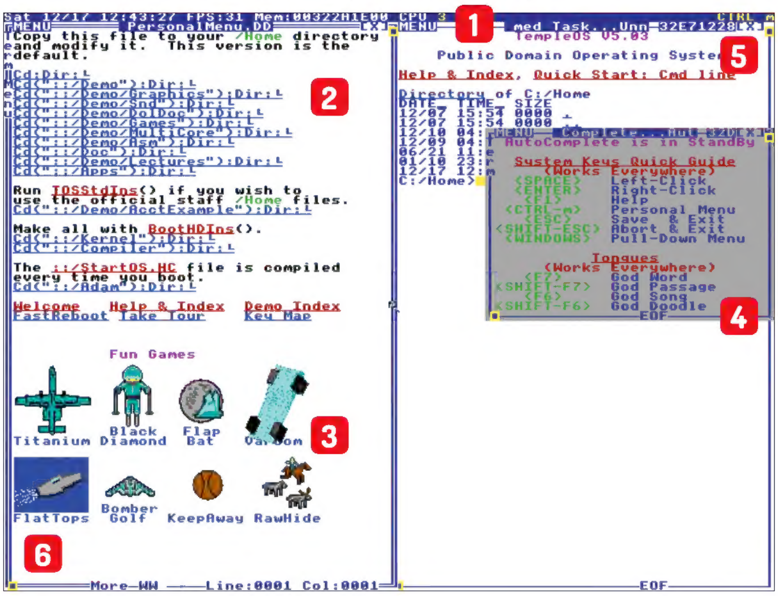
In terms of architecture, what approaches does TempleOS use? Like Linux, it uses a monolithic kernel design, meaning that components such as drivers exist in the same space as the kernel. It's a 64-bit kernel, which has a small performance advantage over older, 32-bit kernels, but more importantly it can access huge amounts of memory and process enormous files and filesystems without having to resort to breaking those resources up into smaller parts to work with them.

As well as being a 64-bit OS, TempleOS has support for multiple CPU cores. From hereon in, the design is a bit unusual for a fairly modern operating system. However, it's worth bearing in mind that TempleOS has an experimental air about it, and it was never designed to be a general-purpose operating system.



Driving game Varoom is one of the more playable example games that come with TempleOS.

## EXPLORE TEMPLEOS



```
Sat 12/17 12:43:27 FPS:31 Mem:00322A1E00 CPU 3                          CTRL m
MENU    PersonalMenu.DD   [X]  MENU     med Task...Unn 32E71228[X]
Copy this file to your /Home directory          TempleOS V5.03
and modify it.  This version is the
default.                                   Public Domain Operating System

Cd;Dir;                                    Help & Index, Quick Start: Cmd line
Cd("::/Demo");Dir;
Cd("::/Demo/Graphics");Dir;                Directory of C:/Home
Cd("::/Demo/Snd");Dir;                     DATE  TIME  SIZE
Cd("::/Demo/DolDoc");Dir;                  12/07 15:54 0000 .
Cd("::/Demo/Games");Dir;                   12/07 15:54 0000 ..
Cd("::/Demo/MultiCore");Dir;               12/10 04:   MENU  Complete...Aut 32D[X]
Cd("::/Demo/Asm");Dir;                     12/09 04:   AutoComplete is in StandBy
Cd("::/Doc");Dir;                          06/21 11:
Cd("::/Demo/Lectures");Dir;                01/10 23:    System Keys Quick Guide
Cd("::/Apps");Dir;                         12/17 12:       (Works Everywhere)
                                           C:/Home>      <SPACE>    Left-Click
Run TOSStdIns() if you wish to                           <ENTER>    Right-Click
use the official staff /Home files.                        <F1>     Help
Cd("::/Demo/AcctExample");Dir;                          <CTRL-m>    Personal Menu
                                                          <ESC>     Save  & Exit
Make all with BootHDIns().                            <SHIFT-ESC>   Abort & Exit
Cd("::/Kernel");Dir;                                   <WINDOWS>    Pull-Down Menu
Cd("::/Compiler");Dir;
                                                              Tongues
The ::/StartOS.HC file is compiled                      (Works Everywhere)
every time you boot.                                       <F7>     God Word
Cd("::/Adam");Dir;                                    <SHIFT-F7>    God Passage
                                                           <F6>     God Song
Welcome      Help & Index   Demo Index               <SHIFT-F6>    God Doodle
FastReboot   Take Tour      Key Map                              EOF

            Fun Games

Titanium  Black Diamond  Flap Bat  Varoom

FlatTops  Bomber Golf  KeepAway  RawHide

More-WW  --Line:0001 Col:0001                                    EOF
```

**1 Status area** This part of the desktop lists things such as the current frame rate of the screen and CPU core utilisation.

**2 File use** When files are listed in a text window, they can be clicked on with the mouse or selected with the cursor keys.

**3 Fun and games** These icons are animated and can be clicked on. Categories include Fun Games, Unfun Games and Not Games.

**4 Shortcut guide** This pop-up window serves as a quick guide to always-available system function keyboard shortcuts.

**5 Windows** TempleOS supports multiple windows that are moved in the familiar way, by dragging the title bar.

**6 See more** If the content of a window is larger than the window, there is a More prompt at the bottom. Scroll down using the cursor keys.

## Multitask multithread

Rather than offering preemptive multitasking like systems such as AmigaOS and Linux, TempleOS uses cooperative multitasking – the applications themselves have to hand control back to the operating system periodically. This leads to multitasking that isn't as smooth, and it presents problems when trying to execute background tasks. It can also result in a system that isn't as reliable, because a crashing app can bring the entire system down. We highlighted this as a shortcoming of RISC OS in a previous article.

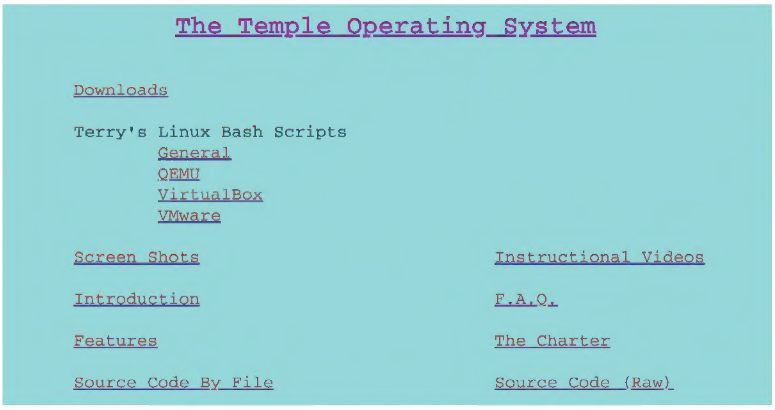
Tasks can launch their own subtasks, meaning TempleOS does, technically, support multithreading. However, it doesn't enjoy the full benefits of that as it doesn't support preemptive multitasking, so threads can't execute independently and simultaneously.

## Memory protection

We expect a modern OS to offer memory protection so user mode programs can't interfere with the memory space of other programs or the kernel. TempleOS

### QUICK TIP

**TempleOS programs are launched by either clicking on them or including them in the current environment: `#include <path and name of program>` loads and executes a program. It's another part of TempleOS that doesn't work like other systems.**



The Temple Operating System

Downloads

Terry's Linux Bash Scripts
General
QEMU
VirtualBox
VMware

| | |
|---|---|
| Screen Shots | Instructional Videos |
| Introduction | F.A.Q. |
| Features | The Charter |
| Source Code By File | Source Code (Raw) |

Some of the TempleOS website is looking worse for wear in terms of layout, but it's still navigable, and it's packed with information and Davis's personal insights.

doesn't work like this. Part of its design ethos is that these restrictions make the system more difficult to program for and to understand. This is in line with the architecture of home computers of the '80s, which were a big influence on the TempleOS design.

On those old home computer systems, the programmer could poke and peek anywhere they wanted, which encouraged experimentation. Indeed, creator Terry A Davis once described TempleOS as a Commodore 64 with multicore x86 support.

The snag is that when you have a system that can run more complex programs, or multiple programs at once, a lack of memory protection could cause havoc. These problems are multiplied when you take a computer online. A lack of memory protection has always been a shortcoming of AmigaOS, to give one example. However, as TempleOS deliberately offers no networking facilities, this latter point is moot.

## QUICK TIP

**Pressing F7 at any time invokes the Tongues function of TempleOS and copies a random word to the screen. Pressing Shift F7 inserts a random Bible quotation. Pressing Escape usually takes you back to the previous screen.**

## Graphics system

TempleOS has a graphics library that can produce 2D and 3D graphics. There is no hardware graphics acceleration, but Davis felt this might be compensated for by the support for multiple CPU cores. As it stands, TempleOS is limited to 640x480 resolution with 16 colours. Unusually, it updates the entire screen 30 times per second and merges the graphics and text layers, creating another barrier to higher resolutions.

## Developing software

Nearly all the software available for TempleOS was created by Davis himself. It isn't a particularly viable target system for ports. The principle programming language of TempleOS, HolyC, is similar to C, but Davis made various customisations. TempleOS doesn't feature any compilers for mainstream languages or frameworks. It is possible to create programs from scratch for TempleOS within TempleOS.

The easiest way to start exploring the example apps included with TempleOS is to press Ctrl+M in a terminal to bring up the Personal Menu. This app launcher can be scrolled up and down using the cursor keys and apps are launched with a mouse click or by pressing Space. As with all of TempleOS, the files and directory listed in this window as text can also be interacted with.

## Creation of TempleOS

TempleOS was largely created by one man, the late Terry A Davis. There's no doubt he was an extremely capable programmer. Unfortunately, he suffered from mental health issues for all of his adult life, and was eventually diagnosed with schizophrenia. He had a degree in mechanical engineering and worked professionally as a VAX programmer, but the mental health issues that dogged him often made it difficult for him to remain employed. At times he was homeless and he had some run-ins with the law.

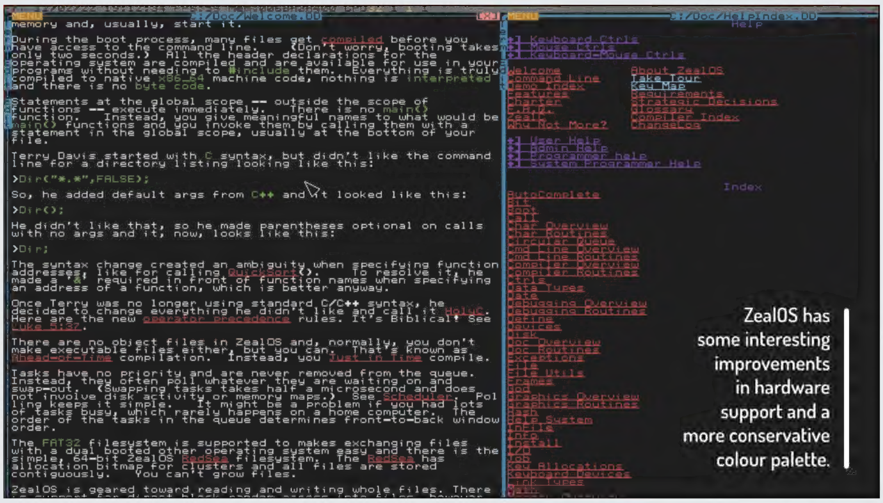
He was a well-known character in the hobbyist OS scene but often fell out with people, and sometimes ended up being banned from online forums. He posted to programming discussion areas of sites such as

## » ZEALOS

ZealOS (**https://github.com/Zeal-Operating-System/ZealOS**) describes itself as "a modernised, professional fork of the 64-bit Temple Operating System" and is a continuation of a TempleOS fork.

Offering notable enhancements over the final version of TempleOS, the first thing that hit us about ZealOS was the more conservative colour scheme. It is also possible to select between multiple resolutions with 32-bit colour depth.

One of the biggest changes is the addition of networking hardware support and a full network stack. This particular addition opens up the possibilities of what the operating system can be used for, but it has to be remembered that the design is inherently unstable and insecure. The hardware support has been expanded with AHCI support for storage devices.



```
memory and, usually, start it.

During the boot process, many files get compiled before you
have access to the command line.   (Don't worry, booting takes
only two seconds.)  All the header declarations for the
operating system are compiled and are available for use in your
programs without needing to #include them.  Everything is truly
compiled to native x86_64 machine code, nothing is interpreted
and there is no byte code.

Statements at the global scope -- outside the scope of
functions -- execute immediately.   There is no main()
function.   Instead, you give meaningful names to what would be
main() functions and you invoke them by calling them with a
statement in the global scope, usually at the bottom of your
file.

Terry Davis started with C syntax, but didn't like the command
line for a directory listing looking like this:

>Dir("*.*",FALSE);

So, he added default args from C++ and it looked like this:

>Dir();

He didn't like that, so he made parentheses optional on calls
with no args and it, now, looks like this:

>Dir;

The syntax change created an ambiguity when specifying function
addresses, like for calling QuickSort().   To resolve it, he
made a '&' required in front of function names when specifying
an address of a function, which is better anyway.

Once Terry was no longer using standard C/C++ syntax, he
decided to change everything he didn't like and call it HolyC.
Here are the new operator precedence rules. It's Biblical! See
Luke 5:37.

There are no object files in ZealOS and, normally, you don't
make executable files either, but you can.  That's known as
Ahead-of-Time compilation.  Instead, you Just in Time compile.

Tasks have no priority and are never removed from the queue.
Instead, they often poll whatever they are waiting on and
swap-out.  (Swapping tasks takes half a microsecond and does
not involve disk activity or memory maps.)  See Scheduler.  Pol
ling keeps it simple.   It might be a problem if you had lots
of tasks busy, which rarely happens on a home computer.  The
order of the tasks in the queue determines front-to-back window
order.

The FAT32 filesystem is supported to makes exchanging files
with a dual booted other operating system easy and there is the
simple, 64-bit ZealOS RedSea filesystem.  The RedSea has
allocation bitmap for clusters and all files are stored
contiguously.   You can't grow files.

ZealOS is geared toward reading and writing whole files. There
```

ZealOS has some interesting improvements in hardware support and a more conservative colour palette.

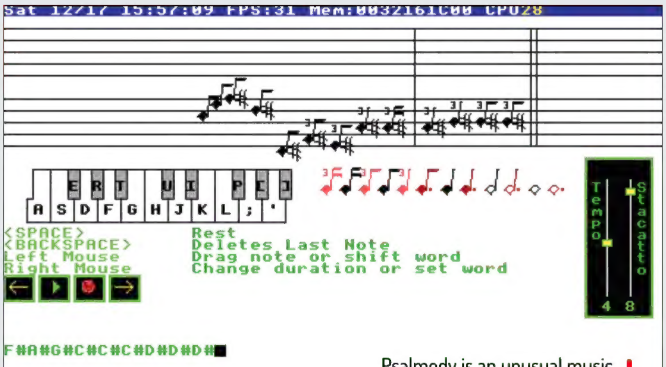
TempleOS is fascinating, but we regard ZealOS as the better starting point if you want to pursue TempleOS's original stated purpose as a "modern Commodore 64" for experimenters and recreational programmers.

## » INCLUDED SOFTWARE

TempleOS has a reasonably complete set of system tools, including those needed to navigate around the filesystem. It has its own filesystem, called RedSea, and it supports FAT32 partitions for file exchange with other operating systems. It also has a plain text editor and a programming editor with syntax highlighting.

The several games that are included with TempleOS are, humorously, divided into the categories of Fun Games, Unfun Games and Not Games. We regard these games as programming examples, and they remind us of the sort of programs that people would type into home computers from listings in magazines in the '80s. That said, they do take some advantage of the power of a modern CPU, as some of them feature smooth scrolling, rotating play areas and 3D graphics.

Within the directories that are referenced on the Personal Menu, there are dozens of smaller graphical examples. It's clear that Davis wanted TempleOS to offer the free and joyful experimental programming fun of 8-bit home computers, such as his Commodore 64 that he originally learned to program on. It's certainly true that, by comparison, modern programming environments tend to get in the way of casual experimentation.



```
Sat 12/17 15:57:09 FPS:31 Mem:00321610C00 CPU28

<SPACE>         Rest
<BACKSPACE>     Deletes Last Note
Left Mouse      Drag note or shift word
Right Mouse     Change duration or set word

F#A#G#C#C#C#D#D#D#
```

Psalmody is an unusual music program, and it's one of a handful of non-game applications.

Reddit, Something Awful, OSDev, HackerNews, Stackoverflow and OSNews, and he posted on social media sites such as Twitter.

Davis's religious beliefs were a major influence. Although there were practical considerations behind the choice of such a low resolution and colour depth, for example, Davis also stated that the decision was due to commandments sent to him from God. He would regularly claim to be in contact with God, and his postings on this matter ranged from aggressive to satirical to incomprehensible. There are some Bible-based functions within TempleOS, and many of the included games are religiously influenced.

His streaming videos and online posts attracted attention from various communities. Many of these people were sympathetic, but some individuals took to cruelly provoking him. There are some saddening videos online of him being provoked by people who'd ring him up on the phone while he was live streaming.

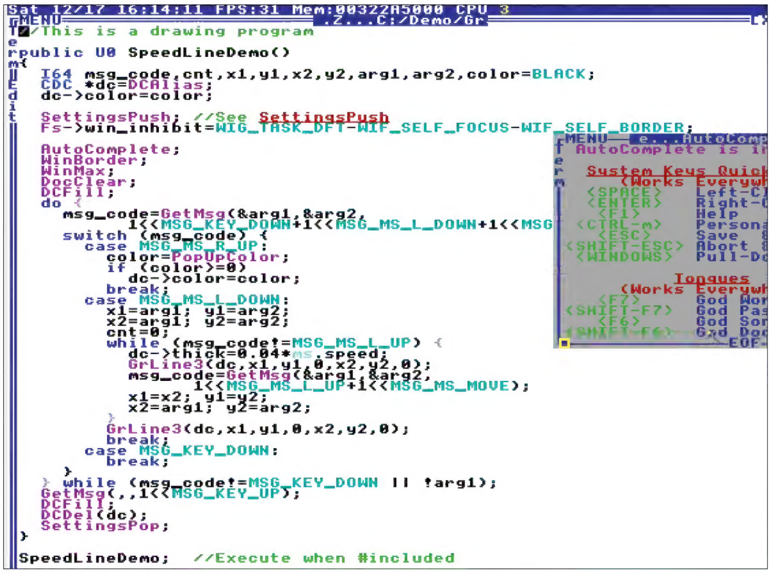
Over the course of 2017, Davis's condition worsened. He video-blogged frequently. In the videos of that era, he is clearly confused, and his speech is strewn with references to bizarre conspiracy theories and frequent racial slurs. He spent most of his last two years living in a van or homeless on the streets. Some of his supporters would donate money to him.

While travelling around the US in 2018, Davis was struck by a train and killed. It is uncertain whether this was a deliberate act of suicide or an accident. Subsequently, various tributes were made by different online communities, mostly in admiration of his technical skills and humour, while sympathising with his mental health problems. TempleOS remains as a final tribute to those skills and his unique way of thinking.

### TempleOS worship

It's clear that TempleOS was never intended to be a primary, general-purpose operating system, and this point was often emphasised by Davis himself. In some ways, this is a more sensible goal than some hobbyist operating systems because, realistically, those systems are unlikely to ever take over from the user's main operating system.

We would recommend exploring TempleOS as a fascinating experience rather than with any particular goal in mind. A continuation, such as ZealOS, is probably a better bet if you intend to use TempleOS for its intended purpose of recreational programming. We have heard of the concept of programming as art, but this is an operating system shaped by the eccentric world view of Terry A Davis. Exploring his operating system is – like trying to understand the man himself – often confusing but constantly both interesting and thought-provoking. LXF



```
Sat 12/17 16:14:11 FPS:31 Mem:00322A5000 CPU 3
MENU                                    .Z...C:/Demo/Gr
//This is a drawing program

public U0 SpeedLineDemo()
{
   I64 msg_code,cnt,x1,y1,x2,y2,arg1,arg2,color=BLACK;
   CDC *dc=DCAlias;
   dc->color=color;

   SettingsPush; //See SettingsPush
   Fs->win_inhibit=WIG_TASK_DFT-WIF_SELF_FOCUS-WIF_SELF_BORDER;

   AutoComplete;
   WinBorder;
   WinMax;
   DocClear;
   DCFill;
   do {
      msg_code=GetMsg(&arg1,&arg2,
            1<<MSG_KEY_DOWN+1<<MSG_MS_L_DOWN+1<<MSG
      switch (msg_code) {
         case MSG_MS_R_UP:
            color=PopUpColor;
            if (color>=0)
               dc->color=color;
            break;
         case MSG_MS_L_DOWN:
            x1=arg1; y1=arg2;
            x2=arg1; y2=arg2;
            cnt=0;
            while (msg_code!=MSG_MS_L_UP) {
               dc->thick=0.04*ms.speed;
               GrLine3(dc,x1,y1,0,x2,y2,0);
               msg_code=GetMsg(&arg1,&arg2,
                     1<<MSG_MS_L_UP+1<<MSG_MS_MOVE);
               x1=x2; y1=y2;
               x2=arg1; y2=arg2;
            }
            GrLine3(dc,x1,y1,0,x2,y2,0);
            break;
         case MSG_KEY_DOWN:
            break;
      }
   } while (msg_code!=MSG_KEY_DOWN || !arg1);
   GetMsg(,,1<<MSG_KEY_UP);
   DCFill;
   DCDel(dc);
   SettingsPop;
}

SpeedLineDemo;  //Execute when #included
```

Practically all of the example programs and system elements are sitting around as source code, ready to be explored.

» **IMPROVE YOUR LINUX SKILLS** Subscribe now at **http://bit.ly/LinuxFormat**